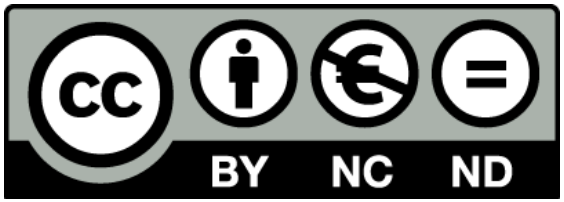
Laboratorio di Programmazione di Sistema

Fondamenti del Linguaggio C Standard

Luca Forlizzi, Ph.D.

Versione 20.1



Perché studiare C Standard in LPS?

- Basso livello di astrazione rispetto alla maggioranza degli altri *HLL*: ciò lo rende estremamente adatto al ruolo di rappresentante dei linguaggi di livello 5
- È un "classico" tra i linguaggi di programmazione, che è opportuno faccia parte del bagaglio culturale di uno studente di informatica
  - il C è studiato o usato (sebbene spesso male) da molte persone, quindi rappresenta una base di conoscenza comune
  - influenza molti linguaggi più moderni in termini di sintassi e semantica, soprattutto a livello di espressioni e *statement*
  - conoscere il C aiuta a capire meglio le architetture hardware e i sistemi operativi
  - a dispetto dell'età e del successo di molti linguaggi più giovani, è tuttora molto usato (si vedano gli indici di popolarità dei linguaggi di programmazione su `www.tiobe.com` e `pypl.github.io`)

Studiare il C: Considerazioni Preliminari

- La portabilità tra diversi ambienti operativi è sempre stata una caratteristica desiderabile di un programma
- Recentemente sono in aumento
  - la diversificazione tra i sistemi
  - il tasso di innovazione
  - la diffusione di software in forma di codice sorgente
- Per questi motivi la portabilità è divenuta estremamente importante
- Per ottenere una buona portabilità è indispensabile conoscere bene gli standard dei linguaggi

Studiare il C: Considerazioni Preliminari

- Numerosi studenti preferiscono creare software da 0 piuttosto che cercare di capire il funzionamento di codice scritto da altri: in contesti lavorativi, spesso la prima opzione non è disponibile
- La diffusione di software open-source e la necessità di sviluppare software di grandi dimensioni fanno sì che a un programmatore sia richiesto di lavorare su software scritto da altri
- Per tali motivi è importante imparare a leggere i programmi
- Anche i programmi scritti secondo stili considerati non efficaci e dannosi in quanto portano alla scrittura di programmi "poco chiari"
- Piacciano o meno, sono stili utilizzati in ambito lavorativo

Studiare il C: Criticità

- Il C viene spesso usato per introdurre le basi della programmazione
  - questa scelta viene fatta in quanto il C, in virtù del suo successo, ha un grande "richiamo" sugli studenti
  - nonostante l'inadeguatezza del C come linguaggio didattico
- La strategia tipica è quella di semplificare e/o omettere dettagli
- Spesso l'esito è negativo
  - gli studenti non sono in grado di usare concretamente il linguaggio C
  - ma invece credono di saperlo fare
  - gli studenti apprendono un linguaggio che è un "C finto"

Studiare il C: Criticità

- "C/C++" **non esiste!**
- Alcuni linguaggi sono fatti in modo che sia possibile usarli efficacemente anche conoscendoli in modo superficiale: il C **no**
- Leggere o scrivere programmi in, C anche molto semplici, richiede una conoscenza approfondita del linguaggio, per evitarne le trappole

Studiare il C: Criticità

- Un sintetico confronto qualitativo delle difficoltà di apprendimento del C rispetto ad altri linguaggi imperativi

Linguaggio	Difficoltà idee base	Difficoltà delle regole	Quantità di nozioni
C	bassa	alta	piccola
C++	alta	alta	molto grande
Java	medio-alta	media	grande
Pascal	bassa	medio-bassa	piccola

- Poiché è un linguaggio piccolo, è facile da ricordare: superato lo scoglio iniziale di impararlo, entra a far parte di un bagaglio culturale che ci si porta dietro in tutta la carriera

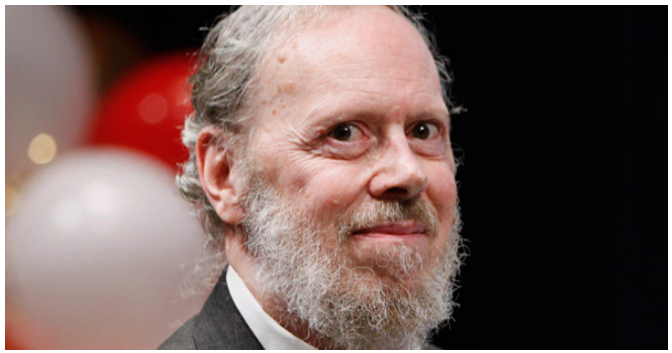
Il creatore del linguaggio C

- Dennis MacAlistair Ritchie nacque il 9 settembre 1941 a Bronxville, New York
- Conseguì un Bachelor Degree in Fisica e un Ph. D. in Matematica applicata dall'Università di Harvard; nella sua tesi di dottorato si occupò di struttura dei programmi e complessità computazionale

Il creatore del linguaggio C

- Si mantenne agli studi lavorando presso il MIT allo sviluppo di computer e software
- Nel 1967 fu assunto da Bell Laboratories, dove lavorò prima nel progetto Multics e poi fece parte del gruppo di lavoro che creò Unix
- Come parte del suo lavoro su Unix, Ritchie creò il linguaggio C
- Tra i suoi lavori successivi, particolarmente degni di nota sono i sistemi operativi *Plan 9* e *Inferno*

Il creatore del linguaggio C



- Nel corso della sua carriera ricevette molti prestigiosi riconoscimenti, tra cui ricordiamo ACM Turing Award (1983), IEEE Hamming Medal (1990), United States National Medal of Technology (1999)
- Dennis M. Ritchie è stato trovato morto il 12 ottobre 2011 nella sua abitazione a Berkeley Heights, New Jersey
- Aveva da poco compiuto 70 anni

Il Contesto

- I linguaggi di programmazione, come quelli umani, evolvono nel tempo. Per capire il funzionamento di un linguaggio bisogna conoscerne la filosofia, e per capire la filosofia è opportuno conoscerne un po' la storia
- Il linguaggio C è un sottoprodotto del lavoro svolto per la creazione del sistema operativo Unix
- Unix fu realizzato nei Bell Laboratories di AT&T, sotto la direzione di Ken Thompson, dopo il sostanziale fallimento del progetto Multics

Il Contesto

- L'hardware per cui Thompson volle realizzare Unix fu il DEC PDP-7 una macchina con 8k parole, ciascuna delle quali formata da 18 bit; una quantità di memoria estremamente limitata, anche per l'epoca
- Per sfruttare al massimo le poche risorse disponibili, la versione iniziale di Unix fu scritta nel linguaggio assembly del PDP-7

Le Origini del C

- Quando Unix fu funzionante, nel 1969, nacque l'esigenza di corredare il sistema di software di utilità e strumenti di sviluppo
- Thompson decise che Unix aveva bisogno di un linguaggio che consentisse di realizzare programmi efficienti sul limitato hardware del PDP-7, ma più comodo da usare rispetto all'assembly
- Nessun linguaggio esistente soddisfaceva questi requisiti, così Thompson decise di inventare un nuovo linguaggio, inizialmente chiamato B
- Lo sviluppo del B non fu pianificato, ma procedette di pari passo all'evoluzione di Unix (di cui era in fase di realizzazione una versione per il nuovo minicomputer PDP-11), adattandosi ai problemi che venivano fuori giorno per giorno

Le Origini del C

- Nel 1971, Dennis Ritchie divenne lo sviluppatore principale (in certi momenti unico) del linguaggio, con un duplice obiettivo
  - estendere il B per dotarlo di un sistema di tipi più sofisticato
  - realizzare un traduttore in grado di produrre codice con velocità e compattezza paragonabili a quelle ottenibili in assembly
- Nel corso del lavoro, il linguaggio venne notevolmente modificato, tanto che Ritchie ritenne opportuno cambiare il nome in C

Le Origini del C

- Agli inizi del 1973 il linguaggio era divenuto sufficientemente maturo ed efficiente (sia in termini di velocità che di quantità di memoria occupata dal codice eseguibile) da consentire di riscrivere in C il kernel di Unix per il PDP-11
- Questo evento fu importantissimo sia per il C che per Unix
  - dimostrò che il linguaggio era potente ed efficiente
  - rese molto più semplice ed economico portare Unix su altre piattaforme hardware, poiché bastava ricompilare il kernel e non riscriverlo tutto in assembly
- Il C divenne immediatamente il linguaggio di programmazione prevalente in Unix

Le Origini del C

- A causa dei problemi con l'antitrust della AT&T, Unix non divenne un prodotto commerciale, ma fu reso disponibile sotto licenza al governo e alle università americane
- La licenza comprendeva la possibilità di accedere al codice sorgente di Unix. Ciò si rivelò utilissimo per le università, in quanto il codice sorgente poteva essere studiato e impiegato per scopi didattici. E naturalmente anche modificato!
- Anche grazie alla disponibilità del sorgente, Unix ebbe un grandissimo successo e una vasta diffusione, e con Unix si diffuse il C

Le Origini del C

- Inoltre, il fatto che fosse possibile realizzare traduttori C molto efficienti e in grado di funzionare con poche risorse, permetteva e rendeva interessante portare il C anche in altri ambienti operativi, persino nei microcomputer (computer destinati ad un solo utente) professionali (PC) o ad uso domestico (home computer) che si diffusero sul finire degli anni 70
- Nel periodo 1973-80 il linguaggio continuò a evolvere, soprattutto in risposta ai problemi di portabilità e di adeguatezza del type system che sorgevano mano a mano che i programmi scritti in C venivano adattati a nuovi hardware

Le Origini del C

- Nel 1978 Brian Kernighan e Dennis Ritchie pubblicarono la prima edizione di *The C Programming Language*, che fu il testo di riferimento sul linguaggio C fino all'introduzione di uno Standard ufficiale; ci si riferisce comunemente al testo con l'abbreviazione K&R
- Nel 1989 viene pubblicato il primo standard del linguaggio C, ad opera dell'ANSI (ente degli U.S.A.)
- A partire dal 1990 l'ISO pubblica lo standard internazionale ISO/IEC del linguaggio C
- La prima versione dello standard ISO è identica allo standard ANSI

Le Origini del C

- Fino alla prima metà degli anni 90, il C è stato molto utilizzato per realizzare sia software di sistema che software applicativo
- Successivamente, è progressivamente aumentata la preferenza per l'uso di linguaggi ad oggetti, nella realizzazione sia di software applicativo in ambienti operativi dotati di interfaccia grafica che di software distribuito, in modo particolare applicazioni web
- Il C è tuttora molto utilizzato nei seguenti ambiti
  - *System programming*: sistemi operativi, sistemi embedded, driver, utilità e servizi di base
  - Compilatori, librerie ed interpreti di altri linguaggi (ad esempio le implementazioni principali di Python, Perl e PHP)
  - Applicazioni in cui è importante l'efficienza: algoritmi, strutture dati, sistemi real-time, software scientifico

Il C è un linguaggio mediocre, nel senso che media tra diverse esigenze

Giuseppe Della Penna

Finalità del C

- Il C non fu progettato "a tavolino" per dimostrare o esemplificare aspetti teorici dei linguaggi di programmazione, e neppure guardando solo alle esigenze dei programmatori
- Il C venne concepito come uno strumento pratico che permettesse di scrivere programmi che svolgessero compiti utili, nel contesto concreto di un ambiente operativo con risorse limitate quale era Unix nei primi anni 70

Finalità del C

- Il C media tra le esigenze degli utilizzatori (i programmatori) e quelle degli implementatori (ovvero i realizzatori dei traduttori)
  - gli utilizzatori avevano l'esigenza di un linguaggio ad alto livello che permettesse di scrivere programmi efficienti
  - gli implementatori avevano l'esigenza di un linguaggio che fosse abbastanza semplice e a basso livello da rendere possibile la creazione di traduttori in grado di produrre programmi efficienti
- È interessante notare che, negli anni 70, utilizzatori e implementatori erano spesso le medesime persone
- In generale, gli utilizzatori erano programmatori molto esperti, in grado di programmare anche in linguaggio macchina, e che capivano bene come i programmi che scrivevano in C si traducevano in codice eseguibile

Requisiti per il C

- I principali requisiti richiesti per il C dal gruppo di sviluppo di Unix
  - avere un livello di astrazione più alto rispetto all'assembly
  - capacità di accedere all'hardware, per eseguire le stesse operazioni possibili con il linguaggio macchina (es: accesso alla porta seriale)
  - disponibilità su un hardware con poche risorse
  - disponibilità in tempi accettabili
- Il primo requisito nella lista tende ad essere in contrasto con gli altri

Requisiti per il C

- In particolare i limiti su risorse e tempo di realizzazione imponevano
  - semplicità di realizzazione di traduttori
  - efficienza del traduttore
  - efficienza del codice eseguibile prodotto
- I requisiti iniziali hanno determinato le scelte fatte da Ritchie e dai suoi collaboratori durante lo sviluppo iniziale del linguaggio
- Molte conseguenze di tali scelte si fanno sentire ancora oggi, nelle regole di base e di dettaglio del linguaggio
- Regole che oggi consideriamo inutili, illogiche o restrittive, avevano una precisa ragion d'essere nel contesto in cui il C nacque e si sviluppò
- L'esigenza di backward compatibility rende difficile modificare le regole del C

Le principali caratteristiche del C

Le principali caratteristiche che contraddistinguono il C rispetto ad altri linguaggi di programmazione, sono le seguenti

- Basso livello (di astrazione)
- Il C è piccolo
- Permissività
- Efficienza
- Portabilità

Basso Livello di Astrazione

Dennis Ritchie, in un celebre articolo presentato alla conferenza HOPL-II, nel 1996, scrive a proposito del C

> *Its types and operations are well-grounded in those provided by real machines, and for people used to how computers work, learning the idioms for generating time- and space-efficient programs is not difficult. At the same time the language is sufficiently abstracted from machine details that program portability can be achieved.*

Basso Livello di Astrazione

- I concetti di base e i costrutti del C non sono troppo distanti dai meccanismi che governano un calcolatore reale
  - il C segue il paradigma imperativo, come il linguaggio macchina
  - operazioni, tipi, istruzioni sono simili a quelle dei calcolatori reali
  - costrutti in C che sembrano semplici non si traducono in codice macchina che richiede una gran quantità di risorse di tempo o di spazio; un programma C semplice e succinto quasi sempre viene tradotto in un codice eseguibile succinto ed efficiente

Basso Livello di Astrazione

- Ma al tempo stesso il C consente
  - leggere e scrivere programmi in modo più semplice, rapido e confortevole di quanto avvenga utilizzando l'assembly
  - di ottenere un buon grado di portabilità
- Il C permette, in alcuni ambiti, di passare attraverso il livello di astrazione della sua abstract machine per accedere ai livelli sottostanti (es: manipolazione dei bit di un valore numerico, accesso a specifiche locazioni di memoria individuate tramite il loro indirizzo); questa capacità è fondamentale per la realizzazione di sistemi operativi

Il C è Piccolo

- Il C è stato realizzato cercando di coprire i bisogni essenziali per ogni programmatore ma cercando di limitare la quantità di concetti e costrutti (evitando il rischio di perdere efficienza e di far aumentare la complessità del compilatore)
- Ha una piccola quantità di concetti base, costrutti sintattici e regole semantiche
- Vi sono pochi modi prestabiliti di fare le cose
- Il C è *compatto*: può essere tenuto a mente interamente

Permissività

Per i programmatori:

- I controlli sulla correttezza sono pochi e semplici
- Il C non impedisce al programmatore di fare ciò che vuole, "si fida" del programmatore
- In generale, il C non cerca di proteggere il programmatore dagli errori che egli può commettere: se il programmatore scrive del codice che a causa di errori fa qualcosa di molto diverso da quella che era l'intenzione del programmatore, il linguaggio non lo impedisce e non fa nemmeno avvertimenti del tipo "Sei davvero sicuro che vuoi . . . ?"

Permissività

Per le implementazioni:

- In alcune circostanze, la semantica di C Standard è definita in modo parziale o non è affatto definita
- Ciò concede margini di libertà che possono essere sfruttati per avere implementazioni più semplici e/o più efficienti

Efficienza

- L'efficienza era uno dei requisiti per il C, ed è stata ottenuta
- Il C è estremamente efficiente, anche su hardware poco potente e con compilatori molto semplici. In tali contesti, per molte applicazioni, ciò è un vantaggio decisivo rispetto ad altri linguaggi
- Su calcolatori più efficienti e grazie alle tecniche di traduzione più moderne, il suo vantaggio in termini di efficienza rispetto ad altri linguaggi è meno significativo

Portabilità

- Grazie al successo della standardizzazione, di solito è possibile portare un programma C da un ambiente operativo ad un altro con uno sforzo contenuto
- Poiché è un linguaggio piccolo e a basso livello di astrazione, è relativamente semplice creare una implementazione del C anche per hardware poco potente, come PC, dispositivi portatili, microcontrollori
- Ha una diffusione quasi universale, è difficile trovare ambienti operativi in cui non vi sia neppure un traduttore C

Altre caratteristiche del C

Le principali caratteristiche che contraddistinguono il C rispetto ad altri linguaggi di programmazione, sono le seguenti

- Flessibilità
- Disponibilità di una Standard Library
- Integrazione con Unix
- Sintassi e stile concisi
- Regole semantiche complesse

Esecuzione di Programmi C

- Un programma C è definito da un testo, chiamato *codice sorgente*, scritto nel linguaggio C
- Un programma C può essere suddiviso in più parti, chiamate *Preprocessing Translation Unit (PTU)*, ciascuna delle quali viene, nella maggior parte delle implementazioni, memorizzata in un *file sorgente*
- Usando il principio di astrazione, possiamo immaginare che un programma C venga eseguito da una apposita abstract machine di livello 5, detta *C abstract machine*

Esecuzione di Programmi C

- L'esecuzione di un programma scritto in C da parte di un dispositivo di livello 2 (abstract machine di una *ISA*) avviene, nella maggioranza dei casi, mediante un processo di traduzione, ovvero costruendo e successivamente eseguendo un programma in *LM*, chiamato *codice eseguibile*
- Il codice eseguibile, eseguito da una abstract machine di livello 2, deve produrre gli stessi output che produrrebbe il codice sorgente eseguito dalla C abstract machine

Esecuzione di Programmi C

- La traduzione di un programma C avviene tipicamente mediante un processo che può essere schematizzato in 4 fasi

1. *Preprocessamento*: viene effettuato separatamente su ogni PTU; consiste in una serie di trasformazioni del codice sorgente, tra cui
   - rimozione dei commenti
   - esecuzione di comandi per il traduttore detti `direttive`

   Ciascuna PTU viene trasformata in una *Translation Unit (TU)*, che è ancora codice C
2. *Compilazione*: viene effettuata separatamente su ogni TU; per ciascuna TU, viene prodotto un insieme di istruzioni *LM*, alcune delle quali non ancora complete, che corrispondono al codice sorgente contenuto nella TU

Esecuzione di Programmi C

3. *Collegamento*: le istruzioni *LM* prodotte dalla compilazione di tutte le TU del programma vengono collegate tra loro e insieme ad alcune librerie utilizzate dal programma; viene prodotto un unico insieme un'insieme di istruzioni *LM*, alcune delle quali non ancora complete, memorizzato come *file eseguibile*
4. *Caricamento*: il contenuto del file eseguibile viene copiato nella memoria principale; vengono completate tutte le istruzioni *LM*, ottenendo il codice eseguibile

- Nella maggior parte dei computer, le prime 3 fasi vengono effettuate, da strumenti di sviluppo software (spesso da un unico traduttore) mentre la quarta dall'ambiente in cui il programma viene eseguito, subito prima di iniziare l'esecuzione

Esecuzione di Programmi C

- L'aggettivo *statico* è spesso usato per indicare che un evento o una proprietà cui l'aggettivo si riferisce accade, o è definito, o è significativo, durante la traduzione di un programma
- L'aggettivo *dinamico* è spesso usato per indicare che un evento o una proprietà cui l'aggettivo si riferisce accade, o è definito, o è significativo, durante l'esecuzione di un programma

Esecuzione di Programmi C

- Durante la traduzione e durante l'esecuzione vengono effettuati dei *checking* o controlli per verificare
  - che il programma sia corretto, ovvero rispetti le regole del linguaggio
  - che il programma, in fase di esecuzione faccia cose "sensate", ovvero che produca un risultato
- Il mancato superamento di un controllo comporta la produzione di un *messaggio diagnostico* (detto più semplicemente diagnostico)

Esecuzione di Programmi C

- I controlli raggiungono il loro obiettivo in modo molto parziale
  - Non riescono a rilevare tutte le violazioni delle regole del linguaggio
  - Molto raramente riescono a capire che il programma contiene delle sequenze di istruzioni che *potrebbero* far si che il programma non produca un risultato corretto
- Un checking particolarmente importante per i benefici che apporta e la disciplina che richiede al programmatore è il *type checking*, che consiste nel controllare la compatibilità tra le operazioni specificate dal programma e i dati cui esse sono applicate

Elementi e Struttura di un Programma C

- Introduciamo rapidamente gli elementi e la struttura fondamentale di un programma C
- Il C è un linguaggio imperativo, e quindi contiene costrutti che
  - definiscono *dati*
  - specificano *istruzioni* per la C abstract machine
  - specificano l' *ordine di esecuzione* delle istruzioni, ovvero in che ordine vengono eseguite le varie istruzioni che compongono un programma
- Uno dei più importanti costrutti è la *funzione*: una funzione è un costrutto che racchiude al suo interno un gruppo di istruzioni, ed eventualmente di costrutti per la definizione di dati, che vengono aggregati in quanto hanno uno scopo comune

Elementi e Struttura di un Programma C

- Un programma C è un insieme (non vuoto) di funzioni, più, eventualmente, alcune definizioni di dato che non sono contenute in alcuna funzione
- L'esecuzione di un programma C inizia con l'esecuzione di una funzione speciale; nei casi che noi considereremo, tale funzione è chiamata `main`
- Le funzioni e le eventuali definizioni di dato di un programma C, vengono inserite in una delle PTU che costituiscono il codice sorgente
- Ogni PTU è, di solito, memorizzata in un file, che ha estensione `.c`, chiamato file sorgente
- La maggior parte dei programmi C che leggeremo e scriveremo in LPS saranno costituiti da una singola PTU

Elementi e Struttura di un Programma C

- La maggior parte dei dati, nei programmi C, vengono memorizzati in entità (contenitori dati) chiamate *variabili*
- Analogamente a quanto accade in Java, ogni variabile ha un *nome* e ha associato un *tipo di dato*, ovvero un insieme di possibili valori che può contenere
- Il nome e il tipo di dato vengono stabiliti dal programmatore mediante un costrutto chiamato *dichiarazione*
- Il tipo di dato di una variabile determina, oltre all'insieme di possibili valori che la variabile può contenere, anche l'insieme di operatori e di istruzioni cui la variabile può essere usata

Elementi e Struttura di un Programma C

- Analogamente a quanto accade in Java, in C è possibile eseguire calcoli con valori costanti e valori memorizzati in variabili, mediante le *espressioni*
- Le espressioni più semplici sono singole variabili o singole costanti: il loro valore è quello del dato contenuto nella variabile o del valore della costante
- Attraverso gli *operatori* si costruiscono espressioni più complesse, il cui valore viene calcolato mediante le regole semantiche di ciascun operatore

Elementi e Struttura di un Programma C

- Il C dispone di un numero elevato di operatori
- Nelle prime lezioni di LPS ci limiteremo all'uso dei seguenti operatori, molto simili a quelli presenti in Java
  - operatori aritmetici
  - operatori di assegnamento
  - operatori di incremento e decremento
- Una delle caratteristiche distintive del C, è che esso enfatizza l'uso di espressioni a scapito delle istruzioni
- L'esempio più significativo è l'assegnamento, che in molti altri linguaggi imperativi non è un operatore ma è un'istruzione

Elementi e Struttura di un Programma C

- Nelle prime lezioni di LPS, studieremo e scriveremo programmi che contengono due soli tipi di istruzione
  - Gli *expression statement*: istruzioni costituite da una singola espressione; l'esecuzione di un expression statement consiste nel calcolo dell'espressione (compresi i suoi side-effects)
  - L'istruzione `return` che ha lo scopo di terminare l'esecuzione di una funzione, eventualmente restituendo come risultato l'espressione che segue la keyword `return`
- Per studiare in dettaglio gli elementi e la struttura fondamentale di un programma C, si rimanda allo studio autonomo dei capitoli 2 e 4 di **[Ki]**

Fondamenti_C: Indicazioni per Studio ed Esercizi

- Per studiare in dettaglio gli argomenti introdotti in questa presentazione
  - Capitolo 2 di **[Ki]**
  - Capitolo 4 di **[Ki]** tranne alcuni contenuti che per il momento si consiglia di tralasciare (verranno presentati nelle prossime lezioni di LPS)
    - i concetti di *undefined behavior* e *implementation-defined behavior* che vengono introdotti nella sezione 4.1
    - i contenuti della sotto-sezione *Order of Subexpression Evaluation* che si trova nella sezione 4.4
- Svolgere il gruppo di esercizi **Fondamenti_C** su *Edu99*